Bezpečnost v IT

3. přednáška

Radek Janoštík

Univerzita Palackého v Olomouci

15. 3. 2024

Outline

Aktuální (kyber)bezpečnostní situace

- Sledujete zprávy z dění v kyberprostoru? Co se stalo?
- Nebezpečné AI modely na `huggingface.co` (Code execution)
- Obejití autentizace na JetBrains TeamCity serveru (Eskalace práv/úplné převzetí)
- Apple vydal bezpečnostní záplaty pro dvě velké chyby (obejití ochrany paměti)
- Joe Biden podepsal výkonné nařízení zakazující prodej osobních údajů do konkrétních zemí
  - Povolení úřadům aktivní obranu
- HW chyba GhostRace – chyba spekulativního vykonávání kódu
- Chyby ve VMWare – možnost opustit virtuální stroj

Kryptografické hashovací funkce

- kryptografická hashovací funkce je zobrazení $\mathrm{hash} : \mathcal{X} \to \mathcal{Y}$, kde:
  - $x \in \mathcal{X}$ je zpráva(soubor) libovolné délky
  - $y \in \mathcal{Y}$ je krátká posloupnost znaků pevné délky nazývaná *hashovaná hodnota* (nebo také *digest* – výtah, souhrn)

Základní vlastnosti

1. hashovací funkce je deterministická
2. hashovací funkce je jednosměrná
3. jediným způsobem, jak získat hashovanou hodnotu $\mathrm{hash}(x)$ zprávy x je vyhodnocení funkce hash pro x
4. lavinovitý efekt (avalanche effect): náhodná nebo záměrná (i velmi malá) změna zprávy na vstupu výrazně změní hashovanou hodnotu

Kryptografické hashovací funkce

Podmínky kladené na bezpečnost hashovacích funkcí

- *odolnost vůči získání vzoru*: pro zadanou hashovanou hodnotu y, je velmi obtížné najít zprávu x tak, že
$$y = \text{hash}(x)$$
- *odolnost vůči získání jiného vzoru*: pro zadanou zprávu x_1 je velmi obtížné najít takovou zprávu $x_2 \neq x_1$ tak, že
$$\text{hash}(x_1) = \text{hash}(x_2)$$
- *odolnost vůči kolizi*: je velmi obtížné nalézt jakékoliv (i náhodné) $x_1 \neq x_2$ tak, aby
$$\text{hash}(x_1) = \text{hash}(x_2)$$

Kryptografické hashovací funkce – analogie

Analogie: otisky prstů

- hashovaná hodnota je ,,otiskem” dat
  - otisk prstu je snadné pořídit
  - máme-li otisk prstu, je nemožné z něj rekonstruovat jeho nositele
  - máme-li konkrétního člověka, je nemožné najít jiného člověka se stejným otiskem
  - je nemožné najít jakékoliv dva lidi, kteří by měli stejný otisk

Shrnuto:

- kryptografické hashovací funkce by se měly chovat podobně jako náhodné funkce
- pomocí hashovaných hodnot umíme identifikovat, vzájemně (rychle, ale!) porovnávat, nikoliv ale zpětně rekonstruovat

Kryptografická hashovací funkce – použití

Ověření integrity souborů:

- po stažení souboru uživatel vypočítá jeho hashovanou hodnotu
- porovnáním s hashovanou hodnotou původního souboru ověří, zda-li nedošlo ke změně

Ověření hesla:

- hesla v databázi nemohou být uložena jako otevřený text (proč?)
- uložíme tedy jejich hashované hodnoty
- počítač je schopen ověřit, zda-li je zadané heslo správné (hashovací hodnoty se rovnají), ale díky odolnosti vůči získání původní zprávy je prakticky nemožné odvodit z hashovaných hodnot příslušná hesla
  - + *sůl*

Digitální podpis:

- Za chvíli

Implementace kryptografických hashovacích funkcí

- MD5 – Message-Digest algorithm 5
  - autor Ron Rivest, rok 1991
  - délka hashované hodnoty je 128 bitů
  - roku 1996 byl proveden *collision attack*
  - nesplňuje tedy podmínku odolnosti vůči kolizi
  - roku 2006 Vlastimil Klíma publikoval efektivní algoritmus pro hledání kolizí
  - MD5 se dnes používá pouze pro ověřování integrity souborů
- SHA-1 – Secure Hash Algorithm
  - navržen NSA (National Security Agency) a publikován NIST (National Institute of Standards and Technology)
  - podobné s MD5
  - délka hashované hodnoty je 160 bitů
  - roku 2005 nalezena slabina
- SHA-2
  - následník SHA-1
  - obsahuje skupinu algoritmů SHA-224, SHA-256, SHA-384 a SHA-512

Hashovací funkce pro hesla

- Většina předchozích navržena i s ohledem na efektivní výpočet pro velké vstupy
  - Např. pro $\approx 5.3GB$ velký ISO soubor trval výpočet SHA-256: 3.6s, SHA-512: 8.8s
- SHA-256 používá Bitcoin $\Rightarrow$ specializovaný HW (ASIC) $\Rightarrow$ miliardy hash/s
  - $\Rightarrow$ Snadné útoky hrubou silou
- Vznik speciálních hashovacích funkcí určený pro hashování hesel
  - Záměrné zpomalení (náročnější výpočet, spotřeba paměti, instrukce bez HW podpory (nebo drahé))
  - Sůl jako další argument
  - Často iterativní výpočet (možno měnit počet iterací)
- Implementace: PBKDF2 (je možné ASIC)
  - scrypt – paměťová náročnost (Litecoin)
  - Argon2
  - Bcrypt – rozšířená, implementace přímo v jazycích/frameworcích

Digitální podpis – motivace

- Klasický podpis – podpisem vyjadřujeme souhlas, autorství
  - Neměl by jít replikovat jinou osobou
  - Sankce při padělání (ukotvení v zákoně)
  - Změna podepsaného dokumentu? (bílá místa)
- Digitální podpis – obdoba klasického
  - S využitím asymetrické kryptografie umí podepsat pouze majitel PK
  - Změna dokumentu po podpisu je očividná
  - Problém v případě prolomení podepisovacího algoritmu (průběžné „přepodepisování“ archivů)

Digitální podpis

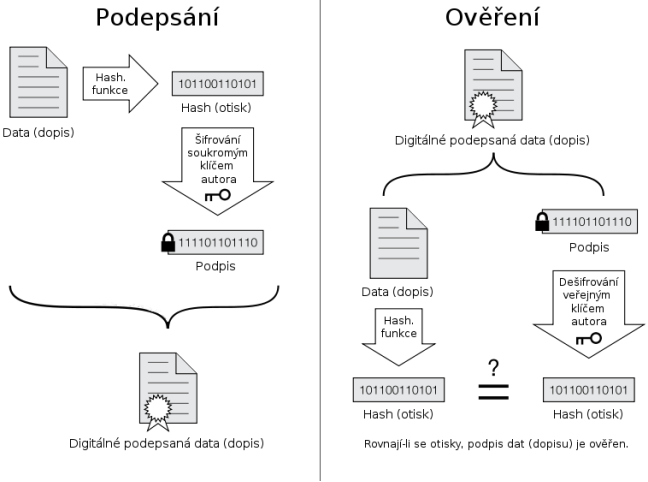
Podepsání:

- Ze zprávy se vytvoří hash
  - Proč ne celá zpráva?
- Hash se zašifruje pomocí soukromého klíče
  - Obměna – je potřeba šifry, která to umožňuje (RSA)
- Zašifrovaný hash se přiloží ke zprávě a je odeslán
  - Potřeba zveřejnit veřejný klíč

Ověření podpisu:

- Příjemce oddělí zašifrovaný hash od zprávy
- Hash se dešifruje pomocí veřejného klíče podepisujícího
- Příjemce porovná hash zprávy a dešifrovaný hash
- Pokud se rovnají, vlastník PK podepsal dokument v daném znění

Digitální podpis



Obrázek: zdroj: Wikipedia

Digitální podpis – podvržení veřejného klíče

- Problém může nastat v distribuci veřejného klíče podepisujícího
- Kdyby byl distribuován spolu se zprávou/souborem mohl by jej někdo po cestě změnit
- *Man In The Middle*(MITM) útok (obrázek na tabuli)
- Problém vyřeší certifikace veřejného klíče „vyšší autoritou“
- Idea: Soukromý klíč bude digitálně podepsán certifikační autoritou
  - Problém: MITM útok „o úroveň výš“
  - Kdo/co je kvalitní certifikační autorita
  - Jakým způsobem soukromé klíče certifikovat?
- $\Rightarrow$ *Public Key Infrastructure*(PKI)

Public Key Infrastructure (PKI)

- Při zavádění asymetrických šifer vzniklo mnoho nekompatibilních norem
  - Nepokrývaly celistvě všechny problémy
  - Aplikace spolu nedokázaly komunikovat
- PKI – soubor norem (organizační i technické), které ošetřují práci s veřejnými klíči a certifikáty
  - Správa
  - Vydávání
  - Platnost
  - Podepisování
  - ...
- Vychází z norem ITU-T řady X.500 avšak není plná oboustranná kompatibilita
- PKI orientována pro použití na internetu (vs. obecnější X.500, SET - platby kartami)

Certifikát

- Definovaná struktura, která obsahuje potřebné údaje pro certifikaci veřejného klíče
  - Identifikační údaje certifikované strany, certifikovaný veřejný klíč
  - Identifikační údaje certifikační autority(CA), sériové číslo certifikátu
  - Doba platnosti, vymezení způsobu použití
- Struktura popsána v *Abstract Syntax Notation One* (ASN.1)
  - Abstraktní jazyk pro popis objektů a jeho vlastností (čísla, řetězce, data, datové typy, . . .)
  - Lidsky čitelná podoba
  - Pro přenos a čitelnost stroji kódováno *Basic Ecoding Rules(BER)* (pořadí bitů, datové typy, . . .)
  - V praxi ještě kódováno Base64 (7bitový přenos) – „PEM formát“
- Ukázka

Certifikát – popis položek (1 / 3)

- Dvě poloviny – data a jejich podpis CA
- Verze – odvozena od verze X.509
  - 1. verze (1988,implicitní) – pouze základní položky
  - 2. verze – položky *issuerUniqueID* a *subjectUniqueID*
  - 3. verze (dnes nejpoužívanější – možnost „custom“ rozšíření
- Sériové číslo – nezáporné celé číslo
  - Jednoznačné v rámci CA (nesmí vydat dva se stejným). Obdoba rodného čísla
  - Nemusí být „popořadě“, záleží na interních pravidlech CA
  - Velikost do 20 bajtů
- Podpisový algoritmus – specifikace, kterým algoritmem je certifikát podepsán
  - Např. `sha384WithRSAEncryption`, `ecdsaWithSHA256`
- Vydavatel – jedinečné jméno CA

Certifikát – popis položek (2 / 3)

- Platnost – omezení z důvodu „trvanlivosti“ šifry, organizační(životnost aplikace)
  - Po vypršení se již nepoužívá k podpisu a k šifrování
  - Pro dešifrování a ověření podpisu i později
  - Případné rozšíření: zkrácení algoritmu k podpisu (certifikát stále platí)
- Subjekt – unikátní jméno – *Distinguished name(DN)* – v rámci CA, obdoba RČ
  - Sekvence relativních jmen – countryName (C), stateOrProvinceName (ST), localityName (L), organizationName (O), commonName (CN), . . .
- Informace o veřejném klíči
  - Použitý algoritmus veřejného klíče
  - Jeho „nastavení“ – RSA – délka klíče, n, e

Certifikát – popis položek (3 / 3)

- Rozšíření – trojice $\langle \text{identifikátor}, \text{kritičnost}, \text{hodnota} \rangle$
  - Kritickým rozšířením musí aplikace rozumět, jinak odmítnout certifikát
  - Authority Key Identifier – CA může mít více klíčů (překrývající se platnost)
  - Key Usage a Extended Key Usage – způsob použití klíče (podpis, TLS, email, . . .)
  - CRL Distribution Points – kde nalezneme seznam odvolaných certifikátů (viz dále)
  - Subject Alternative Name – alternativní jména subjektu – DNS, email

Kvalifikovaný certifikát (QCA)

- Speciální certifikáty, které mají ukotvení v zákoně (v rámci EU)
  - Po právní stránce **nahrazuje ruční podpis**
  - Ošetřuje zákon o elektronickém podpisu (Zákon č. 227/2000)
- Vydáván pouze osobám (ne server, aplikace, . . .)
  - CA „zná“ osobu – ověření oficiálním dokumentem (OP)
- Předmět certifikátu musí být jednoznačný → v případě shody navíc atribut *serialNumber*
- Musí být zajištěna i unikátnost veřejného klíče
  - Archivace veřejného klíče
  - Kontrola unikátnosti při nové žádosti
  - Co dělat při shodě?
- V ČR existují 3 CA pro QCA – PostSignum, 1.CA, eIdentity
- Identifikační údaje: jméno, příjmení, adresa, . . .
- Rozšíření: titul, datum a místo narození, pohlaví, občanství, biometrické informace(kontrolní součty), odkazy na sken podpisu, . . .

Odvolání platnosti certifikátu

- V případě smazání či kompromitace soukromého klíče je potřeba zneplatnit certifikát
- Smazání není tolik kritické (nikdo klíč nezneužije, nevznikne další škoda)
- Kompromitovaný soukromý klíč je potřeba odvolat co nejrychleji – minimalizovat škody
- Máme-li soukromý klíč – mělo by stačit poslat CA podepsanou zprávu s žádostí odovolání
- Nemáme-li (nebo není určen k podpisu)
  - Jednorázové heslo při vydávání (formulář, nepodepsaný mail)
  - Osobně na CA (resp. její registrační autoritu)
- Např. `http://www.postsignum.cz/zneplatneni_certifikatu.html`, `https://sectigo.com/uploads/files/Sectigo_WebPKI_CP_v1_3.pdf`
- Odvolat platnost může i samotná CA
- Kontaktovat druhé strany používání certifikátů (banka)

Certificate Revocation List (CRL)

- CA zveřejňuje seznam odvolaných certifikátů
- Způsob zveřejnění závisí na politice CA
- Obsahuje sériové číslo a datum odvolání
- Zveřejňuje v pravidelných intervalech (vs. přírůstkové CRL)
- Odvolaný certifikát přítomen až do jeho data vypršení
- Seznam musí být podepsán
- Ukázka

Ověření platnosti certifikátu

- Přímá možnost – podíváme se do CRL a ověříme (ne)přítomnost certifikátu
  - Nevýhoda – intervaly vydávání CRL mohou být dlouhé – potřeba ověřit platnost nyní
- $\rightarrow$ *Online Certificate Status Protocol(OCSP)*
- Klient-server protokol pro ověření platnosti certifikátu v reálném čase
- Dotaz: číslo certifikátu, kontrolní součet jména CA a kontrolní součet veřejného klíče CA
- Odpověď: OCSP obálka (nosič pro chyby) + odpověď (kdo, čas, číslo a status certifikátu

Žádost o certifikát

- Tři normy pro žádost o certifikát – RFC-1424(PEM, neujal se), RFC-2314(PKCS#10) a RFC-2511(CRMF)
- Nejčastěji používány – PKCS#10 – vhodný pouze pro certifikáty schopné elektronického podpisu
- Na žádost můžeme pohlížet jako na certifikát podesaný „sám sebou“:
  - Verze žádosti (0, další není)
  - Subjekt – jedinečné jméno, poté uvedeno v certifikátu
  - Informace o veřejném klíči
  - + volitelné atributy (např. challengePassword – jednorázové heslo pro odvolání)
- Podpisem prokážeme držení soukromého klíče (který neopustí naše zařízení)
- Nepodepsané žádosti jsou zahozeny
- Nemusíme komunikovat s CA přímo, často přes Registrační Autority (delegace práce)

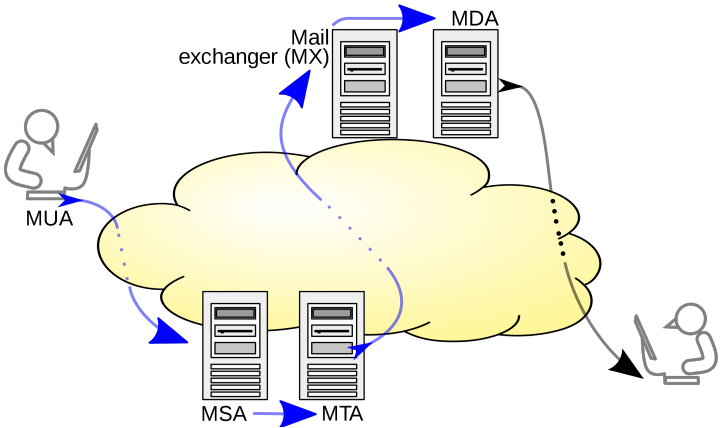
Časová razítka

- Digitálním podpisem pouze deklarujeme pravost dokumentu
- Nikde není uvedeno, že dokument měl takovou podobu v daný čas
- Například u smluv je důležité, kdy byly podepsány
- *Time Stamp Authority(TSA)* – obdoba CA – autorita, která zajišťuje správný čas
- Klient spočítá ze souboru hash
- Odešle žádost o „oražení“ TSA
- Odpověď: Kontrolní součet, unikátní sériové číslo razítka, čas vydání razítka + podpis
- Např: TSA PostSignum
  `http://www.postsignum.cz/casove_razitko_tsa.html`
  - Zpoplatněná služba
  - Nabízí testovací prostředí (klidně vyzkoušejte)

Elektronická pošta

- Elektronická pošta (e-mail) funguje na principech ze 70. let
- RFC-821 *Simple Mail Transfer Protocol(SMTP)* z roku 1982 (nahrazeno RFC-2821 z roku 2001)
  - Uživatelé terminálu mají emailového klienta (lepší textový editor)
  - Umí otevřít složku s poštou, vytvořit a uložit do fronty k odeslání
  - SMTP klient pravidelně prochází frontu a odesílá emaily přes SMTP server
  - Server zjistí, zda je pošta určena lokálnímu uživateli → přesun do složky
  - Není-li, zařadí do své fronty k odeslání
- Návrh počítá s možnou nedostupností serverů (vytáčené spojení)
- Zkouší posílat vícekrát
- Zprávy jsou v textové formě, pouze US-ASCII (7 bitů)
- Servery pro příjem pošty lze najít v MX záznamu v DNS (ukázka)

SMTP v obrázku



Obrázek: zdroj:
`https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol#/media/File:SMTP-transfer-model.svg`

SMTP – Formát zprávy

- Specifikován v RFC-822, nahrazeno RFC-2822(2001)
- Textový formát, zpráva rozdělena na *záhlaví* a *tělo*, odděleno prázdným řádkem (`CRLF CRLF`)
- Záhlaví
  - ASCII-only
  - Tvořeno hlavičkami - `Název: hodnota CRLF`
  - Středník a dvojtečka – oddělovač v seznamu
  - Závorky `<>` v adrese – ignoruje se vše kromě textu v závorkách
  - Hranaté závorky `[]` v adrese – nepřekládání jména
- Základní hlavičky: `From, Sender, Date, Repy-To, In-Reply-To, To, Cc, Bcc, Message-Id, Subject, X-,...`
- Hlavička `Received` přidávají mezilehlé servery (postupně) `from, by, via, with, id, for`

Bezpečnost původního návrhu

- Základní hlavičky nastavuje klient
- Neřeší se jejich pravost – odesílatel, datum – můžeme nastavit cokoliv
- Email prochází internetem v otevřené podobě – kdokoliv po cestě může přečíst
  - Mezilehlé servery mohou i změnit
  - Žádoucí čtení vs. nežádoucí (SPAM filtry)
  - Vztahuje se listovní tajemství na email?
- Původní návrh nepočítá s podepisováním zpráv, přílohami, jiným kódováním
- Podvržení uživatelova DNS (SMTP server, MX záznam)
- Spousta serverů může fungovat bez autentizace (např. omezení pouze IP rozsahu)

ESMTP

- Rozšíření původního SMTP o novou funkcionalitu
- Po „pozdravení“ `EHLO` server vypíše seznam poskytované funkcionality
  - `8BITMIME` – Binární formát, MIME zpráva
  - `SIZE` – Maximální velikost zprávy
  - `SMTP-AUTH` – Odeslání zpráv pouze po autentizaci
  - `STARTTLS` – Možnost šifrované komunikace
- Přidána možnost *Delivery Status Notification* – notifikace o doručení
  - Zpracovává a informuje SMTP server
  - Parametr `NOTOFY` s hodnotami `NEVER, SUCCESS, FAILURE, DELAY`
  - Doručení$\neq$ přečtení zprávy

Poštovní model dnes

- Uživatelé se již nepřipojují na terminál, kde mají poštovního klienta
- Poštovní klient = aplikace u uživatele na PC, přijatá pošta vzdáleně na serveru
- Nutnost stahovat poštu ze serveru
- Protokol *Post Office Protocol(POP)* 3 – v původním návrhu autentizace v plaintextu, později md5, jednoduchý protokol pro stahování pošty ze serveru, textová komunikace (telnet, odchycení, 110/tcp)
- Protokol *Internet Message Access Protocol(IMAP)* 4 – komplexní protokol pro synchronizaci zpráv se serverem.
  - Možnost více zařízení zároveň, textová komunikace 143/tcp
  - Možné stále spojení (notifikace), stahování částí zpráv
  - Vyhledávání na serveru
  - Autentizace heslem (plaintext) nebo jiným mechanismem (Kerberos)
- Oba protokoly mají zabezpečené verze pomocí SSL/TLS – POP3s(995/tcp), IMAPs(993/tcp)

Multipurpose Internet Mail Extension(MIME)

- Omezení na pouze textovou komunikaci je značně limitující –
- Rozšíření hlaviček(RFC-2045-2049)– specifikují typ dat a formát kódování do ASCII
  - `MIME-Version` – Musí být přítomna, stále ve verzi 1.0
  - `Content-Type` – Typ dat v těle zpráv tvar: `typ/podtyp: parametry`. Například: `text/html: charset=utf-8`, `image/jpeg`
  - `Content-Transfer-Encoding` – popis kódovaní do ASCII (quoted-printable, base64, binary, x-)
  - Možné rozšíření pomocí hlavičky `Content-cokoliv`
- Možnost složení zprávy do částí (text, příloha) – `Content-type: multipart/mixed: boundary="lkjassdkl0a2asd"` (obrázek)
- Bezpečnostní problém: Dříve poštovní klienti přistupovali velmi naivně – spouštění příloh, interpretace JavaScriptu, . . .
  - Dnes jsou klienti velmi opatrní
- Pozitivní přínos: Typ zprávy `multipart/signed` a `multipart/encrypted` pro digitální podpisy a šifrování

Bezpečnost přenosu zpráv

- Zprávy putují v otevřené podobě (= kdokoliv může přečíst)
- Nelze ověřit, zda uživatel zprávu opravdu napsal
- Uživatel může popřít, že zprávu odeslal
  - Může prohlásit, že ji někdo po cestě změnil
- Email jako důkaz u soudu?
- $\Rightarrow$Digitální podpis + šifrování

Secure MIME(S/MIME)

- Standard pro šifrování a podepisování MIME dat pomocí asymetrické kryptografie
  - Původní návrh od RSA Laboratories (RSA + MD5 + PKCS#7)
  - Od verze 3 – *Cryptographic Message Syntax(CMS)* (RFC-2630→3852)
- Typy dat
  - `Data` – obyčejná data, bez šifrování
  - `Signed Data` – struktura pro podepsaná data (Povinné: verze, algoritmus kontrolního součtu, podepisovaná zpráva; Nepovinné: Certifikáty ke kořenovému, CRL, elektronické podpisy)
  - `Enveloped Data` – struktura pro šifrovaná data (symetrický klíč – zašifrován pro každého adresáta, kombinace soukromý odesílatele a veřejný příjemce (Diffie-Hellman), symetrický klíč šifrován symetrickým klíčem z předchozí zprávy
  - `Digets Data, Encrypted Data, Authenticated Data`
- Zpráva v CMS vložena jako MIME data do emailu (typy: `application/(x-)pkcs7-mime`)
- Hlavička `From: (Sender:)` musí souhlasit s certifikátem podpisu

S/MIME bezpečnostní úskalí

- Pouze podepsaná zpráva obsahuje nezašifrovaný původní text
  - Možná triviální změna „po cestě“
  - Odchytnutý email můžeme změnit, úplně z něj odstranit podpis a podpisové hlavičky
  - Uživatel si může myslet, že odesilatel jen zapomněl email podepsat
- Použití závorek <> v adresátovi, ale jiného textu před nimi
  - Někteří poštovní klienti zobrazovali pouze jiný text, nikoliv adresu
  - Zpráva byla podepsaná, ale vypadala jako od někoho jiného
  - Dnes (naštěstí) většina klientů zobrazuje data z podpisu
- Využití testovacích CA pro generování falešných (ale někde důvěryhodných) certifikátů
- Většina webových klientů nepodporuje S/MIME (nedešifrovaní zpráv, nemožnost podepsat)

Protokol SSL/TLS – motivace, vývoj

- Transportní vrstva v modelu TCP/IP neřeší zabezpečení komunikace
  - Mezilehlé uzly mohou data číst (hesla, komunikace)
  - Inteligentní útočník může pozměnit data
  - Komunikující strany mohou popřít části komunikace
- Potřeba zabezpečit aplikační data
- Protokoly *Secure Socket Layer(SSL)* a *Transport Layer Security(TLS)*
  - Dva nekompatibilní protokoly
  - TLS vychází z SSL v3.0
  - TLS v1.0 je *de facto* SSL v3.1
- SSL vyvinula firma *Netscape* (Microsoft měl obdobný protokol – PCT)
  - Verze 1.0 kolem roku 1994 – nepublikována
  - Verze 2.0 (1995) – mnoho zranitelností, problémy při použití, „zastarána“ RFC-6176 (2011)
  - Verze 3.0 (1996) – brzo nahrazena TLS 1.0, *Internet Engineering Task Force(IETF)* – RFC-2246

Protokol SSL/TLS – náhled

- Slouží k zabezpečení dat z aplikačního protokolu – „mezivrstva“ (obrázek)
  - Nijak nezkoumá, nemodifikuje aplikační data
  - Zabezpečí(a zkomprimuje), přenese a „rozbalí“
- Architektura klient/server
  - Obousměrná komunikace (různé šifrovací klíče)
  - Server se vždy prokazuje certifikátem
  - Autentizace klienta certifikátem volitelná (anonymní režim)
- Zajišťuje šifrování, integritu a autorizaci(kontrolní součet dat a sdíleného tajemství)
  - Aplikační data rozdělena na fragmenty
  - Šifrování symetrickou šifrou (hybridní šifrování)
  - Data nejsou elektronicky podepsána
- Struktury popsány ve vlastním jazyce (podobnost s C), nepoužit ASN.1, BER či DER

Struktura SSL (1 / 2)

- SSL se „navenek“ tváří jako jeden protokol
- Je složen ze 4 dílčích protokolů
- *Record Layer Protocol(RLP)*
  - Zajišťuje manipulaci s daty z aplikační vrstvy
  - Komprimuje, šifruje, počítá kontrolní součty
  - Rozbaluje, dešifruje, kontroluje integritu
  - Neřeší tvorbu klíčů, stanovení algoritmů, ...
- *Handshake Protocol(HP)*
  - Navazuje komunikaci mezi stranami
  - Zajišťuje dohodu na šifrovacích, kompresních a hashovacích algoritmech
  - Autentizuje strany
  - Výměna data pro výpočet *hlavního tajemství*, odvození šifrovacích klíčů
  - Výpočet sdíleného tajemství pro kontrolní součet(*MAC secret*)
  - Příprava budoucí *protokolové svity*
  - Přenášen pomocí RLP jako aplikační protokol (zajímavost)

Struktura SSL (2 / 2)

- *Change Cipher Specification Protocol(CCSP)*
  - Jednoduchý, jeden účel (jasný z názvu), jediná zpráva
  - HP nastavuje budoucí svitu
  - Strana komunikace oznamuje „přecházím na novou svitu“
- *Alert Protocol(AP)*
  - Servisní protokol pro oznamování závad
  - Obdoba ICMP protokolu z vrstvy IP
- Obrázek SSL komunikace

Zranitelnosti SSL

- *Padding Oracle On Downgraded Legacy Encryption(POODLE)* – MITM útok (2014)
  - Mnoho serverů podporovalo SSL 3.0 pro zpětnou kompatibilitu
  - Prostředník se *dočasně* vydává za server a donutí uživatele udělat downgrade na SSL 3.0
  - Poté, využije zranitelnost SSL 3.0 (nepočítá hash se zarovnáním), zkouší možnosti, server správná data přijme
  - Zjištění jednoho bajtu na maximálně 256 pokusů
- Útoky na kompresi – The Compression Ratio Info-leak Made Easy(CRIME) – 2012 a he Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext(BREACH) – 2013
  - Hádání částí zpráv na základě prodloužení/zkrácení zprávy kompresním algoritmem
- Útoky na zastaralé, ale podporované protokolové svity
  - Výchozí nastavení serverů nemusí po čase reflektovat bezpečností „trendy“
  - Správci by měli reagovat a již nebezpečné svity nepodporovat
  - Nástroje na kontrolu – např. `https://www.ssllabs.com/ssltest/`

Heartbleed (2014)

- Chyba nebyla v protokolu, ale v jeho implementaci – OpenSSL
- Chyba v rozšíření *heartbeat*, které slouží k udržování spojení
- Klient posílá udržovací zprávu tvaru *délka* + *data*
  - „Jestli tady jsi, pošli mi: *4B – ahoj*“
- Server odpovídá kopií zprávy
  - „4B – ahoj“
- Klient pošle špatnou délku
  - „Pošli mi: *65536B – ahoj*“
- Server odpověděl zkopírovanou zprávou + obsah paměti dané délky
  - V paměti mohly být citlivé informace (PK, klíče pro šifrování, cookies, . . .)

Protokoly používající SSL/TLS

- „SSL verze“ aplikací mají speciální port nebo možné „povýšení“ na šifrovanou verzi
  - Např. HTTP port `80/tcp`, HTTPS port `443/tcp`
  - „Povýšení“ pomocí hlavičky `426 Upgrade Required`
- HTTPS
  - URI: `https:` – DNS jméno či IP adresa musí být v Alternativním jméně předmětu v certifikátu
  - Možné více jmen(subdomén), případně *wildcard* – `*`
  - Možné dvě úrovně autentizace – SSL/TLS(certifikáty) a HTTP Basic Auth (jméno a heslo), lze kombinovat
- POPS a IMAPS
  - Rozšíření o povýšení v RFC-2595
  - POP3 – příkaz *STLS*, IMAP4, ESMTP – příkaz `STARTTLS`
  - „pevné porty“ – POP3s 993/tcp, IMAP4 995/tcp, SMTPS 465/tcp

Doporučená četba

- Schneier B. *Applied Cryptography Second Edition*. John Wiley & Sons, 1996. ISBN 0-471-12845-7.
  - Kryptografické hashovací funkce
- Dostálek L. a kolektiv. Velký průvodce protokoly TCP/IP: Bezpečnost (2. aktualizované vydání). Computer Press, 2003. ISBN 807226849X
  - Kapitola 3 – MIME
  - Kapitola 10 – S/MIME
  - Kapitola 6 – ASN.1, BER, DER
  - Kapitola 7 – PKI
  - Kapitola 11 – SSL/TLS
- Dostálek L., Kabelová A. Velký průvodce protokoly TCP/IP a systémem DNS (5. aktualizované vydání). Computer Press, 2008, ISBN 978-251-2236-5
  - Kapitola 17 – Elektronická pošta
- Zákon o elektronickém podpisu - Zákon č. 227/2000
  - `https://www.zakonyprolidi.cz/cs/2000-227`